Bachelor's thesis

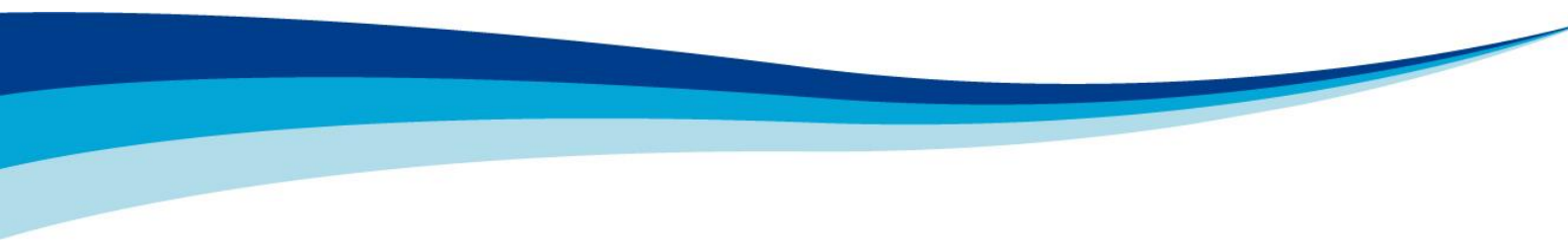
Degree programme: Information Technology

Specialisation: Internet Technology

2014

Bidyottama Koirala

DEVELOPING AN XML-BASED APPLICATION



TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

BACHELOR´S THESIS | ABSTRACT
TURKU UNIVERSITY OF APPLIED SCIENCES

Bachelor of Engineering | Information Technology

June, 2014 | 54

Ferm Tiina

Bidyottama Koirala

DEVELOPING AN XML-BASED APPLICATION

The volume of data in the World Wide Web has expanded dramatically over the last few years. Efficient methods of accessing the data, sharing and extracting information have become some of the critical needs of today's web. Conventional tools prove to be useless with these issues. The web, therefore, needs to evolve from HTML with the help of the latest revolutions in structural and content modelling techniques which introduce XML (EXtensible Markup Language), Topic Maps, RDF (Resource Description Framework) and other similar technologies.

This thesis explores the practical application of XML as a potential approach to make the web and content management system more semantic and powerful. The main aim of this thesis is to build a system which relies on an XML-database instead of the traditional database models.

Different XML-related technologies like XML Schema, DTDs (Document Type Definitions) XQuery, XSLT (EXtensible Stylesheet Language Transformations) are used in the process which contributes towards developing a data system that relies on XML databases and views over them.

As a part of the thesis, a simple application of XML which demonstrates the strong practical features of XML is developed. The final outcome of the thesis project is an XML-based XHTML (EXtensible HyperText Markup Language) page.

KEYWORDS:

XML, HTML, XHTML, Databases

CONTENTS

FIGURES

TABLES

LIST OF ABBREVIATIONS (OR) SYMBOLS

XML	EXtensible Markup Language, is a metamarkup language which defines rules for encoding documents
HTML	HyperText Markup Language, the predominant markup language for web pages.
DTD	Document Type Definition, defines the legal building blocks for an XML document
XQuery	A query language that has been designed to query and transform structured or unstructured data, mostly in the form of XML
XSLT	EXtensible Stylesheet Language Transformation, is a language used to transform XML into other forms like HTML, plain text, etc
XHTML	EXtensible Hypertext Markup Language is a family of XML that extends the versions of HTML.
WYSIWYG	“What You See Is What You Get” a term used in computing, which implies a user interface that allows the user to view the document similar to the end result while the user is creating the document.

1 INTRODUCTION

The quantity of data in the World Wide Web is constantly expanding at a rapid rate [1]. Effectively organizing, managing, sharing and retrieving of data are essential functions of a document management system. However, the limitations of current technology have made it difficult to manage the vast amount of data. A number of businesses are trying to make their presence felt in the web but are hindered because of the incompatibilities with the legacy data systems. XML was designed to handle all these limitations and is intended to serve as a one of the technologies upon which the future web could be built upon.

This thesis provides the reader with an understanding of what XML really is, its history and the importance of XML in context of World Wide Web. The application which is developed for Turku University of Applied Sciences as a part of the thesis project provides the readers familiarity with the practical application of XML as a data model.

The application thus developed demonstrates the strong capabilities of XML, allowing readers to understand the practical implications of XML. The output of the application is an XHTML web page which is built over an XML-database.

The thesis starts with a brief introduction of XML, its history, importance and the anatomy of XML documents in Chapter 2. Processing and validating XML documents with various tools is the subject of Chapter 3. Different XML-related technologies that are essential in developing a XML based system are discussed in Chapter 4. Chapter 5 covers step-by-step the process of practically developing an XML-based application. The results of this process are documented and concluded in Chapter 6 and 7, respectively.

2 XML

Extensible Markup Language or XML is a meta-language for any kind of information, a data storage toolkit, an open standard that has enthralled the imagination of technology masters. X in XML stands for EXtensible, which means that the language can be extended, adapted and tailored according to need. It should be noted that despite its name, XML is not itself a markup language. It is a set of rule for developing markup language.

XML is a standard endorsed by W3C for document markup. It provides a generic syntax used to markup documents with tags, which is both human and machine readable. It offers a standard format for computer documents. This format is flexible and can be customized to meet the needs of diverse domains. [2]

2.1 Emergence of XML

The world has taken a giant leap in information technology in the twentieth century. The amount of information has increased at an exponential rate. Organizing all this information has become a major issue of concern.

During the last half of this century, it was very difficult to write programs to search for and draw the information, cross reference it electronically or adapt documents for different applications. This was because earlier electronic formats have mainly focused on the presentation aspect of documents rather than on the meaning and structure of documents. Two early formatting languages troff and TeX could not provide logical structure or semantics to the documents. Generic coding was later developed to solve these limitations of formatting codes. Graphic Communications Association (GCA) was the first organization to put forth this idea of using descriptive tags and generic codes. A project called “Gencode” was developed during late 1960s to encode documents with generic tags.

Later IBM launched a project which created Generalized Markup Language, (GML) for solving the issue of encoding the documents for various information

subsystems. This language used content-based tags, which facilitated editing, formatting and searching of documents by different programs.

The American National Standards Institute (ANSI), influenced by the success of GML formed a team which would later develop a standard text-description language based on GML. Consequently, Standard Generalized Markup Language (SGML) was created. SGML was then extensively applied by the military, aerospace and publishing industries.

SGML was developed as a toolkit for creating specialized markup languages. SGML is flexible in its design and functionality. The major drawback of SGML is that it is very complex and possesses a lot of esoteric parameters. Software that is built to process SGML can be complex and expensive.

However, Hyper Text Markup Language (HTML), is a limited application of SGML which offers only few of the functionalities of SGML. HTML was in some way a reverse step since some principles of generic coding had to be abandoned to attain the necessary simplicity. For example, HTML allows users to use only a finite set of tags, which are fashioned to represent web pages in presentational manner. So, it cannot really function as more than a conventional markup language. It would not be possible to use HTML to exchange data between incompatible databases. Therefore, to apply the principles of generic coding to the Web, it was attempted to adapt SGML to the Web. Nonetheless, this proved to be a complicated task because SGML was too huge to compress into a little web browser. Therefore, an indispensable need for a smaller language which preserved the functionality and generality of SGML was felt. Hence, was born EXtensible Markup Language, XML [1].

2.2 Importance of XML

Descriptive markup

XML permits the users freedom of creating their own markup language for specific purpose. This is an advantage because users can have unlimited number of markup languages specially tailored according to need. Users can define their own tags and construct data which is self-descriptive and easy to understand. Figure 1 is an example of a very simple but complete and self-explanatory XML document.

```
<?xml version="1.0" encoding="UTF-8"?>
<person>
<name>Albert</name>
<address>244 Wiseman Street</address>
<city>San Diego</city>
<state>CA</state>
<zip_code>92123</zip_code>
</person>
```

Figure 1. A simple XML document

Use of meta-data

Meta-data means simply “data describing data”. Data is usually useless without metadata information. XML has a standard method and syntax for encoding the meaning of data, metadata. This provides a method for encoding the semantic information of data. Table 1 demonstrates the difference between data and meta-data referring to the Figure 1 above.

Data	Meta-data
Albert	Name
244 Wiseman Street	Address
San Diego	City
CA	State
92123	zip_code

Table 1.Comparison of data and meta-data

Data Portability

XML offers potential for sharing data across different platforms. The basic goal of XML is to encode the documents in a way that the data is transportable from one environment to another retaining the purity of original data. XML is a really simple, structured, and straightforward data format. Since XML documents are text, any tool that can read text file can read XML documents. [2, p. 6]

Figure 2 represents the portability of XML documents across different platforms.

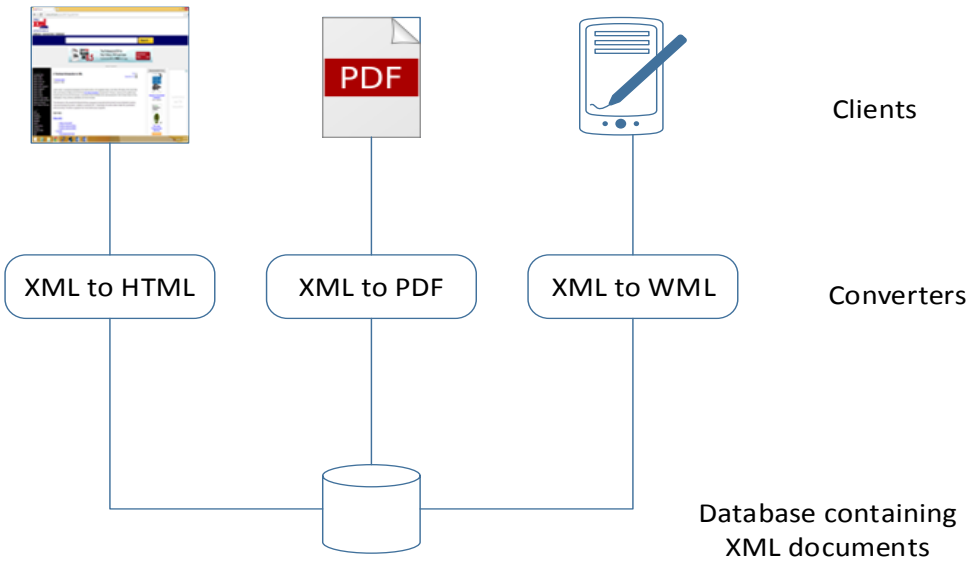


Figure 2. Organizing and sharing information with XML

Unambiguous structure

Even though XML is flexible in the elements it allows to be defined, it is strict in most of its other aspects. XML expects users to follow a set of rules that is pre-defined. These rules restrict the users to mark documents up in such a manner that there cannot be any ambiguity in the interpretation of names, order or hierarchy of elements. This helps in the minimization of possible errors and reduction in the complexity of codes. Parsers can easily interpret the data and do not have to try to guess and fix the errors in the documents.

Users are also free to create rules on how the documents should look. The Document Type Definition (DTD) and XML schemas are the tools that aid in this process.

Separation of stylesheets from main documents

XML allows the author to keep the style information separate from the XML document, called stylesheet. This has several advantages compared to having style information in the same XML document. The tags in the XML document are descriptive about their function within the document rather than just describing how documents should look. The segregation of meaning and style is important in XML documents. The documents which do not count on stylistic markup can be easily repurposed and converted into new forms.

Since it is possible to apply multiple stylesheets to documents, different versions of documents can be created by applying a different stylesheet, which makes it possible for the same document to be viewed and printed on several devices without modifying the original XML document. [1]

Advantage over conventional database models

Traditional database systems like the entity-relationship data model have been around for a long time. These database systems have been very efficient in storing highly structured and normalized data. However, it is evident that the data in the real world is not normalized or squared. These items of data are found to be mostly semi structured. For example, let us consider a customer list where there is variation among the data of different customers like a customer has an email address and a home phone number and another customer has a home phone number and a work phone number. A typical database model represents each property as a column on a relational table. This approach causes the table to have sparse entries and denormalized data. This can lead to problems in scalability and performance. These kinds of problems can be overcome by using XML databases instead of the traditional database models. [3]

Other important benefits of using XML over traditional databases include:

- Representation of data in XML is more natural and logical than conventional data structures.
- Relational databases force the users to consider data objects into flat relation tables and the relationships between them as key relationships. Nevertheless, in reality the data objects are hierarchical and cannot be considered flat. XML expresses the relationships between objects or elements as hierarchical tree.
- XML databases are independent of the platform they are used in but conventional databases are not.
- In most of the cases, databases are coupled with schema tightly. However, XML separates the code from the schema, which makes it easier to modify the database schema as needed. [3] [4]

Infrastructure for the Semantic web

XML can be considered one of the infrastructures upon which semantic web can be built upon. The semantic web is the second generation of the web, and the extension of current web in which information has well-defined meaning. The idea behind the semantic web is that web pages should be machine-understandable instead of being only human-readable. It is the collection of XML documents, semi-structured databases, and other objects in the web. [5]

XML provides the syntactic layer for the Semantic Web. XML produces semantically rich documents because of its use of meta-data and focuses on the meaning of documents rather than the presentation. In other words, using XML helps us to add meaning to the web data by adding the metadata and these meaning can be “understood” by computers. So it can be implied that XML is the first level of the semantic web. [4] [6]

Open Standard

XML is a standard that is not a proprietary of any company or is not associated with a particular software. [7]

2.3 Anatomy of XML Documents

XML Prolog

A prolog is the initial line in a XML document. It contains an XML declaration. Typically, an XML declaration contains an XML version number and the encoding used. In the example the version number is 1.0 and the encoding used is UTF-8. UTF-8 is also the default for documents without encoding information.

```
<?xml version="1.0" encoding="UTF-8"?>
```

XML expects that all XML software understand UTF-8 and UTF-16 encoding. Most XML software also understand encodings like ISO-8859-1, Windows-1252, and ASCII. [8]

XML Tree

The structure of XML documents can be compared to a tree. Documents start at the “root” and braches to the “leaves”. The documents themselves define the structure [9]. Figure 3 depicts an example XML document that has a `person` element that contains information which is marked up suitably to present its meaning.

```
<?xml version="1.0" encoding="UTF-8"?>
<person born="26/12/1791" died="18/10/1871">
    <name>
        <first_name>Charles</first_name>
        <last_name>Babbage</last_name>
    </name>
    <occupation>Philosopher</occupation>
    <occupation>Mathematician</occupation>
    <occupation>Inventor</occupation>
    </person>
```

Figure 3. An example XML document

The above example XML document can be represented graphically as a tree (Figure 4). The tree, in this example starts at the `person` element and branches and leaves out to various elements.

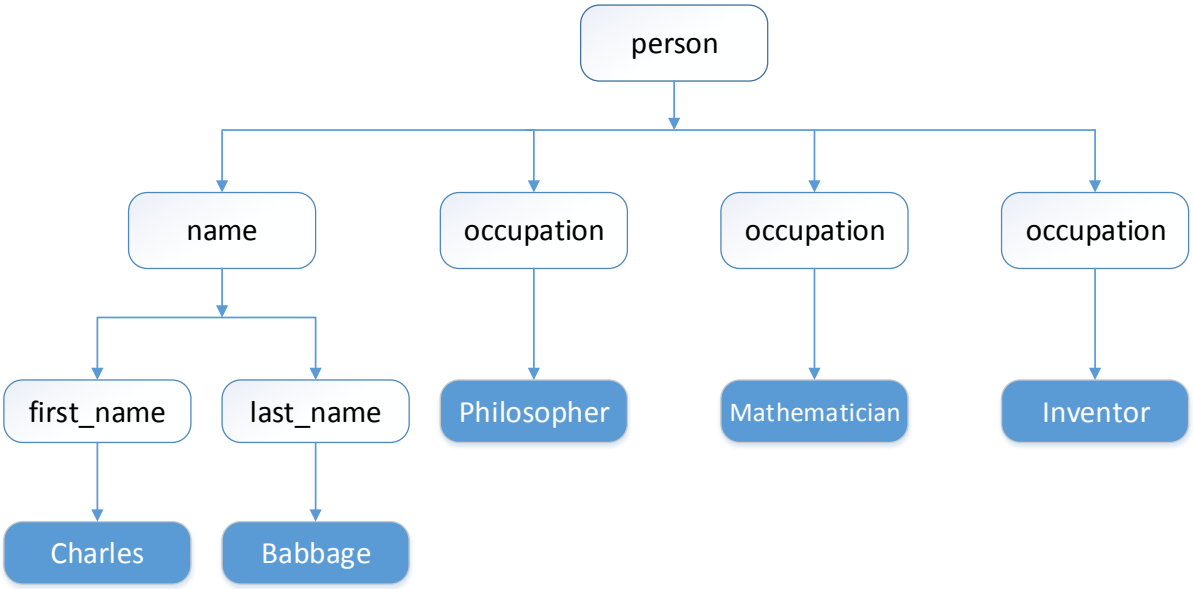


Figure 4. Tree structure of Figure 3

Parents and Children

Figure 4 is composed of one person element. The person element again contains four child elements: one name element and three occupation elements. The name element has two child elements: first_name and last_name. The person element is the parent of the name and occupation elements. name is the parent for first_name and last_name.

Root element

In an XML document, one element contains all the other elements. This element exists without a parent. In Figures 3 and 4, the person element satisfies the criteria and is called the root element.

Elements

An element describes and encapsulates the data that it contains. An XML element can contain other elements, attributes or text. [10] Each white box in Figure 4 represents an element and each blue box represents character data.

Attributes

XML attributes are the name-value pair connected to the element's start tag. Name and value are separated by an equals sign and potentially optional whitespace [11]. Values can be inserted inside single or double quotation marks. In the example XML document in Figure 4, the `person` element has the `born` attribute with the value 26/12/1791 and `died` attribute with the value 18/10/1871.

```
<person born="26/12/1791" died="18/10/1871">
```

3 PROCESSING AND VALIDATING XML DOCUMENTS

This chapter focuses on the practical aspect of creating XML documents. It further progresses discussing the various elements in processing the XML documents. The process of parsing XML document is discussed in brief and various requirements for validity of these documents are explained.

3.1 Processing XML Documents

An XML document is generally created with an editor. This editor can be a basic text editor, like Notepad, that cannot comprehend XML at all or a WYSIWYG editor, like Xeditor, or an advanced structured editor, like XML Marker that displays XML documents as trees. An XML document is frequently a document in computer's hard disk but is it not always necessarily so. An XML document can be a record, an extract from a database or a torrent of bytes obtained from a network. [12]

The document thus created is processed by a tool called XML processor. The XML processor acts as a bridge between XML documents and the application which will use the XML document in the end. The possible number of XML processors is unlimited. XML processors can be anything ranging from XML editors, validity checkers or web browsers etc.

A parser is a fundamental and significant component of an XML processor. It reads and transforms XML documents into an internal representation for other programs and procedures to use. Document parsing is a very important and an expensive operation that can upgrade or degrade XML processing performance. There are three steps for parsing an XML document. Figure 5 is the graphical representation of the following steps.

1. The XML file is read by an XML parser.
2. Parser converts the stream of character into byte-sized tokens.
3. An abstract representation of the document (like an object tree) can be assembled in the memory using the tokens. [1, p. 11]

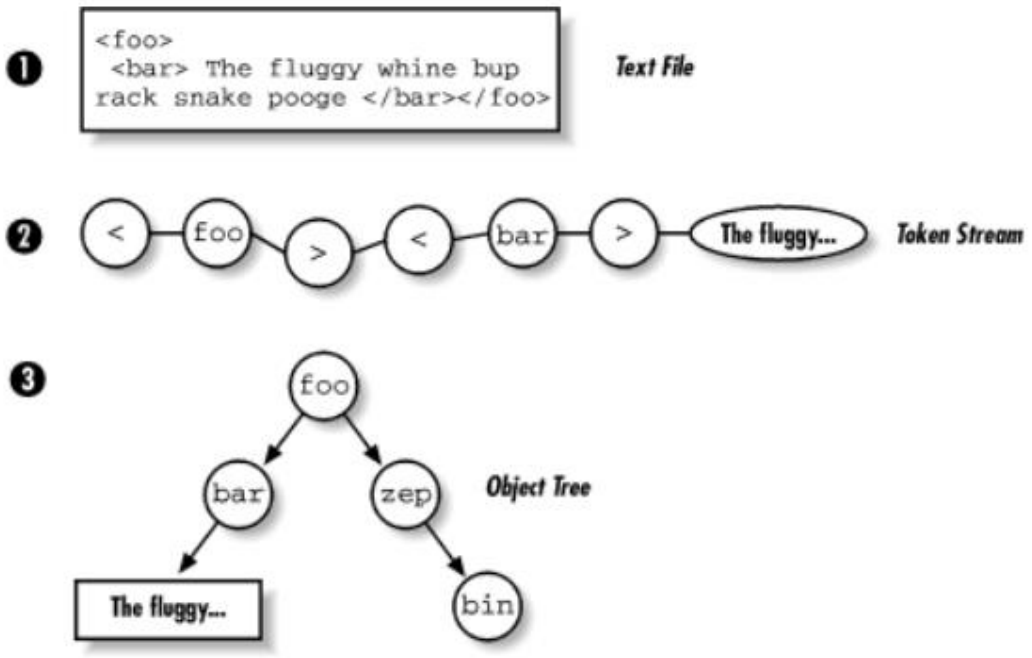


Figure 5. Steps of parsing an XML document [1, p. 11]

The major responsibility of the parser is to divide the document into elements, attribute, and character data and pass the individual piece of information to the application. When a parser recognizes a possible error in the document, it aborts the parsing process and reports the error to the application. However, some parsers may read past the first error in the document, continue parsing and detect other errors. Nevertheless, they will no longer pass the contents of the documents to the application.

XML parsers are known to be very strict and precise about the rules that an XML document must follow. They do not ignore even the most trivial of errors like missing of an end tag or inconsistency in the characters used in the start and end tag. The logic behind XML parsers being so specific in their operation is that they want to make the functioning of XML documents as predictable as it can possibly be. Since documents are meant to be used universally, work

everytime in the same manner, they must satisfy the syntax rules and be well formed. This picky nature of XML parsers in turn increases efficiency and reduces the possible frustration that might occur when hunting down the bug.

Some rules that XML documents must satisfy for well-formedness are as follows:

- A non-empty element must be delimited with a start and an end tag.
- The start and end tags must be matching and consistent.
- An empty element's tag must have "/" before the end bracket.
- Attributes must be quoted using single or double quotes.
- Elements should not overlap. They must be properly nested.
- There should be a single "root" element which contains all the other elements.
- The name of elements can only start with letters or underscores. They can only contain number, letters, periods, hyphen and underscores. [1, p. 51]

A syntactically correct document is called a well-formed document. However, some rules can also be specified later using a Document Type Definition (DTD). Checking a document against constraints in a DTD is called validation. Even though validation is an optional step in XML processing, a valid document should confirm to the rules specified in the DTD.

3.2 Validating with Document Type Definition (DTD)

XML is a flexible meta-markup language and it can be utilized to write domain-specific application. Even though it is so flexible, all the applications cannot afford such flexibility. For example, it would be least expected to find an element named G_Clef inside a biology document since elements like G_Clef is to be found in documents related to music. This kind of rules can be specified explicitly using a DTD. [2, p. 7]

A Document Type Definition or DTD is specification of permitted elements, attribute, entities and some possible constraints in their combination. A document which fails to confirm to the DTD will be called invalid. However, the document can still be called a well-formed XML document. An XML document which violates the constraints in the DTD can still remain an XML document.

When a validating parser detects an error, it reports it to the application which will use the XML document. The application then decides whether or not to continue with parsing the document. The validity error is not necessarily fatal, unlike a well-formedness error.

A document can contain the DTD itself, or a pointer called system identifier which gives the URI reference of the externally located DTD. [13] The benefit of storing DTD externally is that it allows to be easily referenced to multiple XML documents.

Figure 6 is the `person` XML document which now contains DTD in itself.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE person[
<!ELEMENT name (first_name, last_name)>
<!ELEMENT person (name, occupation*)>
<!ATTLIST person
   born CDATA #FIXED "26/12/1791"
   died CDATA #FIXED "18/10/1871"
>
<!ELEMENT last_name (#PCDATA)>
<!ELEMENT first_name (#PCDATA)>
<!ELEMENT occupation (#PCDATA)>
]>
<person born="26/12/1791" died="18/10/1871">
```

Figure 6. An example XML document containing DTD

The DTD in Figure 6 is interpreted as:

- !DOCTYPE person defines that the root element of the document is person.
- !ELEMENT name defines that the name element contains two elements: first_name and last_name.
- !ELEMENT person defines that the person element contains two elements, name and occupation. The element occupation can occur in multiple instances.
- !ATTLIST person defines that the person element has two attributes, born and died both of which have fixed values.
- !ELEMENT first_name defines the first_name element to be of the type "#PCDATA" .
- !ELEMENT last_name defines the last_name element to be of the type "#PCDATA" .

A document type declaration with an externally located DTD declaration looks like Figure 7. In this example, the externally located DTD is referenced to the XML document.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE person SYSTEM "person.dtd">
<person born="26/12/1791" died="18/10/1871">
```

Figure 7. Referencing an externally defined DTD

3.3 XML Schema: An alternative to DTD

It has been a very frequent complaint that DTD uses old, inflexible and inexpressive syntax. Besides that, documents use one syntax and DTDs use another, which is rather strange. A need for a more powerful tool for defining the structure of an XML document was felt, which led to development of the XML Schema (XSchema).

XML Schema, like DTD, also defines the structure of an XML document. XML Schema provides a means of defining the content and semantics of a document. XML Schema is a description of XML document, which imposes constraints on the structure and content of documents. However, XML Schemas add more functionality than DTDs. Everything that a DTD can define can be defined with XML Schema but not vice-versa. [1, p. 153]

Here is a list of some advantages of using XML Schema over DTDs: [14]

- XML Schema has support for set of data types like the ones used in most other programming languages. These types are much broader than the basic ones like `CDATA` and `PCDATA` in DTD.
- XML Schemas also provide additional functionality to create additional data types.
- XML Schemas are written in XML syntax, so they are XML documents, which means it is not necessary to learn a new language to write XML Schemas. Another benefit of this feature is that they can be edited using XML editors and parsed with standard XML parsers.
- Inheritances for attribute, element and data type definitions are supported by XML Schema.
- DTDs enforce basic grammatical rules for defining XML document according to the meta-data which constitutes the structure of document. Whereas XML Schema provides elaborate rules to define what the data can and cannot contain. [15]

Figure 8 presents an extract of XML Schema of the `person` XML document.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="person">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="name"/>
        <xs:element name="occupation" maxOccurs="3"/>
      </xs:sequence>
      <xs:attribute name="born" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="26/12/1791"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="died" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="18/10/1871"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>
```

Figure 8. XML Schema

This XML Schema can be interpreted as:

- <?xml version="1.0" encoding="UTF-8"?> is the declaration of the schema document. The version of XML used is 1.0 and the encoding is UTF-8.
- <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"> In this line of the code, the root of the XML Schema is defined as <schema> and the fragment xmlns:xs="http://www.w3.org/2001/XMLSchema" denotes that the element and data types used in schema come from "http://www.w3.org/2001/XMLSchema" and the namespace should be prefixed with xs:.
- <xs:element name="person"> defines the element called `person`.

- <xs:complexType> indicates that the element is a complex type.
- <xs:sequence> indicates that the complex type is a sequence of elements.
- <xs:element name="name"/> defines another element name which is under person element.
- <xs:element name="occupation" maxOccurs="3"/> defines another element occupation which can occur maximum 3 times.
- <xs:attribute name="born" use="required"> defines an attribute born and that it is a required value.
- <xs:simpleType> defines that the attribute is a simple type.
- <xs:restriction base="xs:string"> restricts that the attribute can only be a string.
- <xs:enumeration value="26/12/1791"/> specifies that the only acceptable value for the attribute is "26/12/1791".
- <xs:attribute name="died" use="required"> defines an attribute died and that it is a required value.
- <xs:restriction base="xs:string"> restricts that the attribute can only be a string.
- <xs:enumeration value="18/10/1871"/> defines that the only acceptable value for the attribute is "18/10/1871".

The externally located XML Schema can be referenced in the XML document. An example of how the schema can be referenced in the person XML document is presented in Figure 9.

```
<?xml version="1.0" encoding="UTF-8"?>
<person born="26/12/1791" died="18/10/1871"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="person.xsd">
```

Figure 9. Referencing XML Schema

The first action to take while referencing XML Schema is declaring in the document the XML Schema Instance namespace which provides access to the attributes required in referencing XML Schemas. The prefix mapping xsi: and the URI associated is added. Doing this produces the code fragment xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance".

Adding the location for XML Schema to use for the namespace's constraints is the next step. The XML Schema of the `person` element does not use namespaces which results in the code fragment xsi:noNamespaceSchemaLocation="person.xsd". [16]

4 XML RELATED TECHNOLOGIES

Different XML-related technologies, like XPath, XQuery, XSLT, etc, are used in the process of creating XML applications. This section provides an overview of these technologies and their roles in building XML applications.

4.1 XPath

XPath is developed by World Wide Web Consortium (W3C) to navigate through XML documents. XPath is named so because of its use of path notation to navigate through the hierarchical tree of XML documents. Even though XPath is designed to be used for XML documents, it uses a compact non-XML syntax. The operation of XPath depends on the abstract logical structure of documents rather than the superficial syntax. [17]

XPath considers each XML document as a tree of nodes. The nodes can be root, elements, attributes, processing instructions, comments, etc. XPath also considers the relationship between the nodes for example parents, children, siblings, ancestors, and descendants. [18]

XPath by its very design is meant to be embedded in other host languages such as XQuery or XSLT.

4.2 XQuery

XML has taken over the world of technology in recent years. The Internet extensively relies on XML database and file system for the storage of information. Structured data, such as, spreadsheets or unstructured data such as images and videos are now being stored in XML.

These data are used for a wide variety of purposes. Different elements of these data can be subject of interest for different fields. For example, a sales report can be used for financial statements, or by tax authorities to create tax reports or the report can be used for future planning. Therefore, to meet these

requirements, W3C designed a language which could query and transform the XML data according to need.

XQuery is a query language which allows users to select the XML data components of interest, rearrange and transform them and return the output in the desired structure. XQuery is also sometimes called “SQL of XML”.

Uses and Capabilities

XQuery can

- select desired information according to specific criteria
- filter out unwanted data
- manipulate and transform data
- extract information from relational databases to use in a web service
- aggregate data from multiple sources
- restructure XML data into another form
- extract information from XML databases and present the output [19]

Processing queries

An example of basic XQuery processor model is shown in Figure 10.

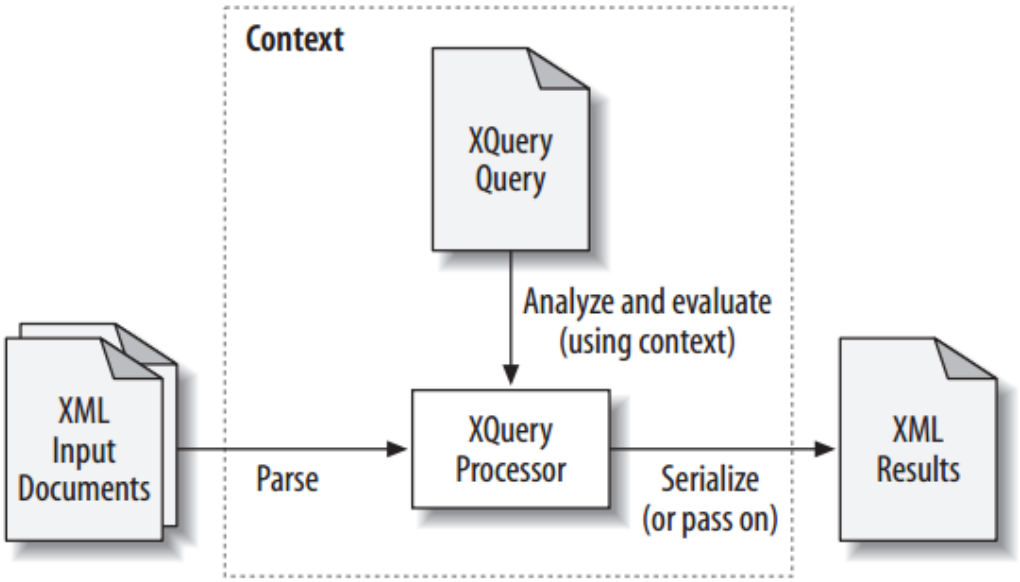


Figure 10. A basic XQuery processor [19, p. 15]

In this case, an XML input document is any XML data source that is being queried. It can be any kind of XML document such as text files that are written in XML, data stored in XML databases, extracts of XML documents that are retrievals from the Web using a URI.

The XQuery query can be located in a text file or it can be a piece of code dynamically generated by a program. The query can also be embedded inside a program code or a query library.

The query is evaluated in an environment which we refer here as context. The context includes all the elements which affect the evaluation of the query. The context can include values such as date, time and time zone.

The XQuery Processor is a software or a component of software that parses, analyses and evaluates the query. The analysis and evaluation are analogous to compilation and executation of a code. At the analysis stage, the XQuery Processor looks for syntax and other errors that are independent of the original input data. At the evaluation phase, it evaluates and accesses the output of the query based on the input XML documents.

The results are then returned as a series of values by the Query Processor. The results can be written to an external XML document, sent to a user interface or conveyed to another application. The process of writing the results to an XML document is called serialization. [19, pp. 15-17]

4.3 XSLT

XSLT stands for EXtensible Stylesheet Language Transformations, which is a style sheet language for XML documents. XSLT is a subset of XSL (EXtensible Stylesheet Language) and is a recommendation of W3C.

XSLT is a flexible language used to transform one XML document into another form of document such as an HTML, XHTML or PDF. XSLT also helps in

adding, removing, rearranging elements and attributes to and from an output document. [20]

XSLT uses XPath to find and navigate different elements in an XML document.

Processing with XSLT

XSLT documents contain specific rules according to which an XML document is to be transformed to another one. XSLT documents contain stylesheet templates. The XSLT processor compares the elements in the source XML document with the stylesheet template. Upon finding a matching template, it writes the contents in the template to the output document. After completing this step, the XSLT processor may serialize the output into an XML document or can produce the output in another format such as HTML. This process is graphically illustrated in Figure 11. [2, p. 129, 21]

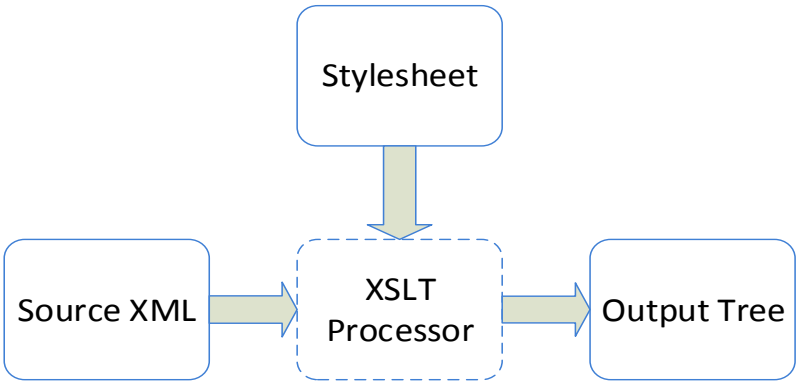


Figure 11. Processing XSLT transformation

4.4 XHTML

Extensible Hyper Text Markup Language (XHTML) is a newer generation of HTML which is modularized and reformulated to comply with the standards of XML. It has been a W3C recommendation since January 2000. [22]

HTML has only offered limited features which are mostly presentational. Unconventional data contents, such as musical notations or chemical formulae, were difficult to handle using HTML. HTML also cannot keep up with newer demands of organizing, processing and distributing data. [23] A number of these increasing demands can be satisfied by XML-compliant languages such as XHTML which combines the strength of HTML and the power of XML. [24]

The relationship between SGML (on page 7), XML, HTML and XHTML is shown in Figure 12.

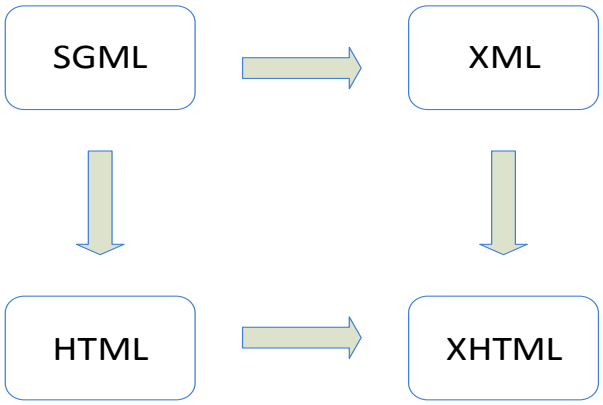


Figure 12. Relationship between SGML, XML, HTML and XHTML [6]

XHTML documents are written in XML syntax, which require much more discipline than writing HTML documents. Some basic rules for writing XHTML documents which are different from HTML are:

- Elements must be nested properly.
- Every start tag must have a corresponding end tag.
- Unlike HTML, XHTML is case sensitive.
- Each attribute in XHTML must be quoted.

5 DEVELOPING AN APPLICATION BASED ON XML

To demonstrate the application of XML in practice, for the purpose of this thesis an application based on XML is developed. This application is developed for Turku University of Applied Sciences. Different XML related tools and technologies are used in the development process.

5.1 Creating XML database

This XML application can be created using different XML editors, like XMLSpy, ExchangerXML, Liquid XML Studio, etc. Altova's XMLSpy is chosen for the development of this application. It is downloaded and installed from Altova's website. [25]

To start with the development process, first a new XML document is created. After declaring the XML version and specifying the encoding used, a root element is created. Suitable and self-descriptive tags are used. For this XML database/document, the root element is <TUAS>. The <TUAS> element has a child element, <Branches>. <Branches> has four child elements namely <Joukahaisenkatu>, <Sepänkatu>, <Salo> and <Lemminkaisenkatu> which are derived from the physical locations of the different branches of Turku University of Applied Sciences. Each of these elements has the descendants <People>, <Studies> and <Others>. Each <People> element has <Staff> and <Student> as children. <Staff> and <Student> each contain basic information about staffs and students like <first_name>, <last_name>, <email> and <address>. The <Studies> element contains <Courses> and <Research> as child elements. The <Courses> element contain information about different courses, like <name>, <credits>, <date>, <time>, <instructor> and <place>. The hierarchical structure of this document can be seen more clearly in the XML Schema diagram in Figure 15.

The XML document thus created is shown in Figure 13. Elements higher in the hierarchical order are shown in the figure and are not expanded completely.

Therefore, most of the descendent elements remain hidden. The complete XML document is attached in the Appendix.

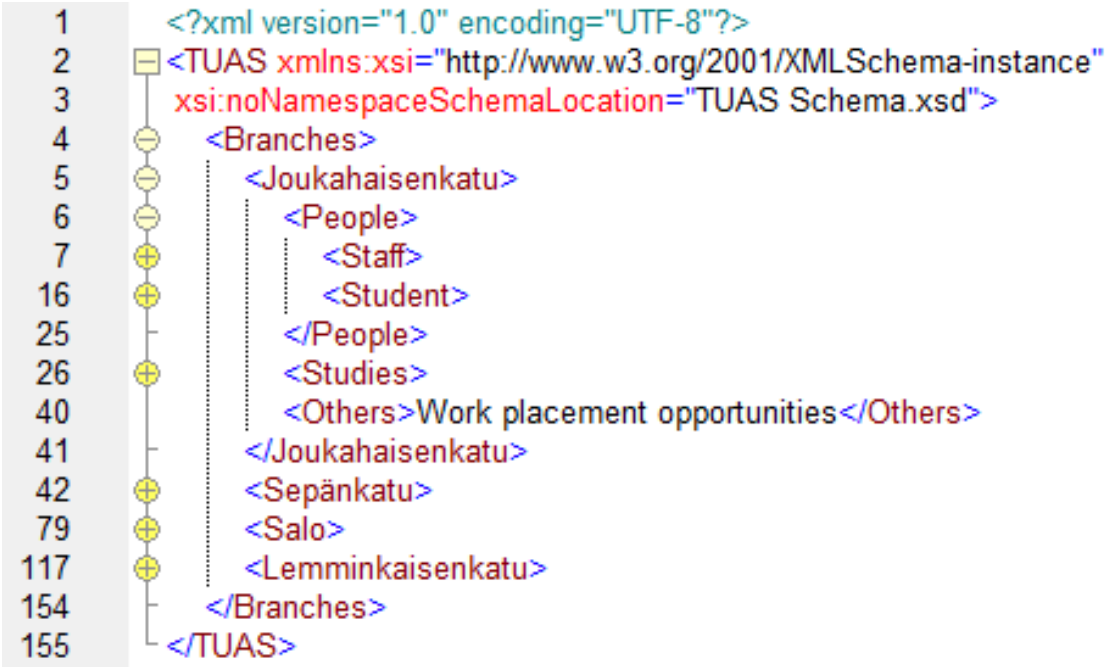


Figure 13. TUAS XML-database

The created XML document is then checked for any well-formedness errors.

5.2 Defining the constraints: Creating XML Schema

This step can also be completed before creating the XML database and an instance of XML database can be created from the XML Schema. For this application though, the constraints are defined after creating the XML document.

The root element of the schema document is <schema> and it is declared that the elements and data types used in the schema come from the "http://www.w3.org/2001/XMLSchema" namespace. Elements that contain other elements or attributes, like <TUAS>, <Branches>, <Salo>, <name> are defined as <complexType>. Other elements which do not contain further elements or contain only data for example, are specified as <date>, <time>,

<email>, <group>, <place>, <address>, <first_name>, etc are defined as <simpleType> elements. These elements contain some character values, so they are also restricted to being only strings. Element <credits> is restricted as a byte. A snippet of the XML Schema is shown in Figure 14.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
3    <xs:element name="Salo">
4      <xs:complexType>
5        <xs:sequence>
6          <xs:element ref="People"/>
7          <xs:element ref="Studies"/>
8          <xs:element ref="Others"/>
9        </xs:sequence>
10     </xs:complexType>
11   </xs:element>
12   <xs:element name="TUAS">
19   <xs:element name="date">
20     <xs:simpleType>
21       <xs:restriction base="xs:string">
22         <xs:enumeration value="01.01.2014-31.07.2014"/>
23         <xs:enumeration value="09.12.2013-14.04.2014"/>
24         <xs:enumeration value="22.04.2014-06.06.2014"/>
25         <xs:enumeration value="28.08.2013-15.12.2013"/>
26       </xs:restriction>
27     </xs:simpleType>
28   </xs:element>
29   <xs:element name="name">
```

Figure 14. XML Schema of TUAS XML document

The graphical representation of the XML Schema generated with XMLSpy is shown in Figure 15.

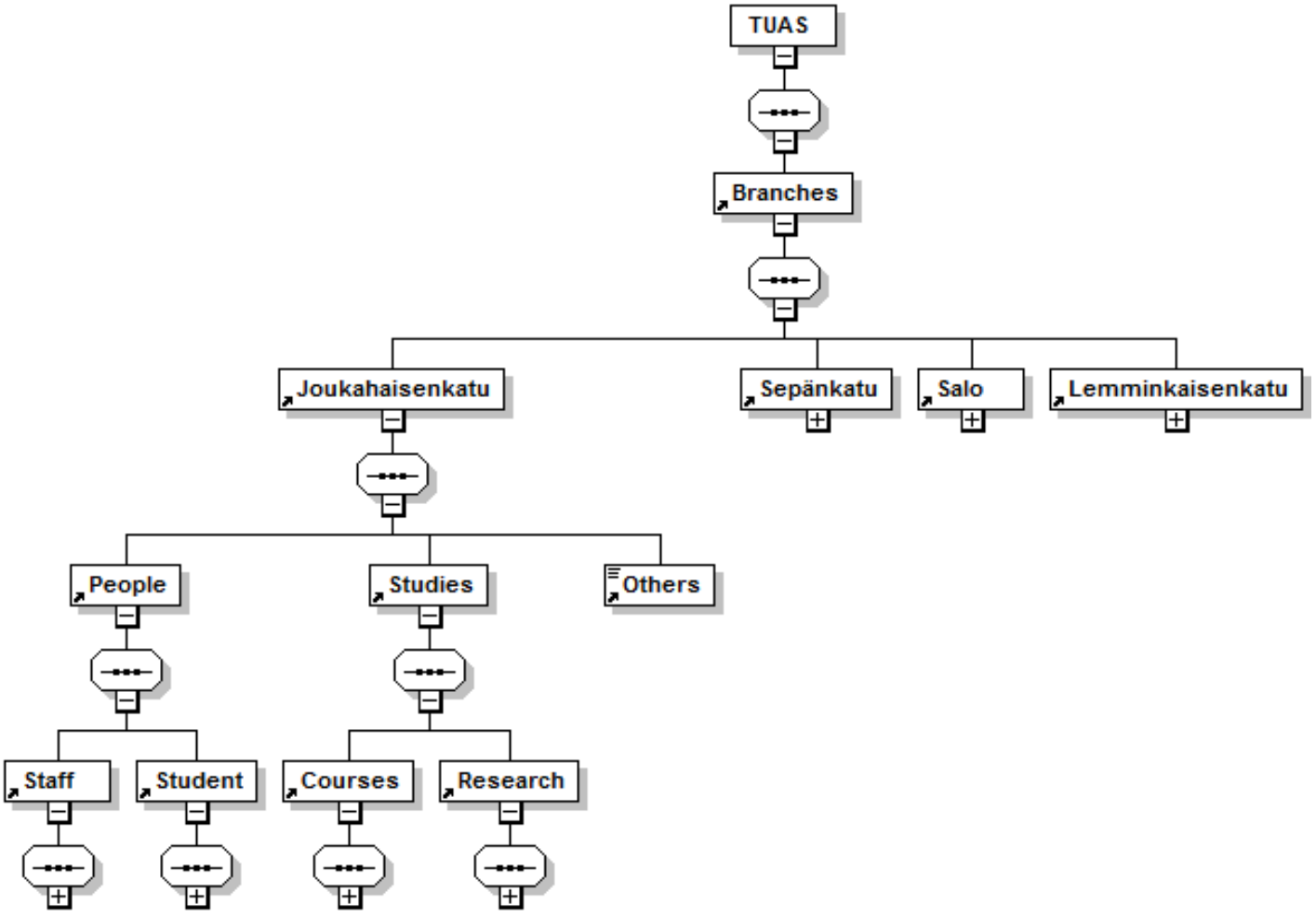


Figure 15. Graphical representation of TUAS XML Schema

It is necessary to link the XML Schema document to the original XML document. The following code fragment

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

is added to the root <TUAS> which gets the XML Schema Instance namespace available to be used. After that the schemaLocation attribute is used to specify the location of XML Schema for the <TUAS> XML document. The code

xsi:noNamespaceSchemaLocation="TUAS Schema.xsd"

specifies the location of the XML Schema created.

The <TUAS> XML document is then checked for validity against the XML Schema specified.

5.3 Creating web-page content: Querying the database

XQuery is used here to create web-page content as a view over the underlying XML database already created. For this application, we are interested in extracting all the names and other information of staffs and students belonging to each branch of TUAS from the main XML document, and displaying the results as a web-page. A simple XQuery is defined for this purpose.

XQuery version 1 is used. XPath syntax is used to navigate to the target nodes <Staffs> and <Students>, for example:

```
TUAS/Branches/*/People
```

This syntax is embedded inside the XQuery document. The complete XQuery document is attached in the Appendix.

The query is then executed and applied to the XML document. The name and information of the staff and students are returned as an output. This output is then stored as an XML file.

5.4 Transforming and displaying the results as a webpage

The result from the query, which is an XML document, can be then transformed with the help of XSLT. XSLT allows an XML document to be translated and styled into XHTML, PDF or other forms. Alternatively, Cascading Style Sheet Language (CSS) can also be used to specify style information for these documents.

A new stylesheet XSLT file is created. Then, XSL version 2.0 is declared. XSLT namespace:

xmlns:xsl="http://www.w3.org/1999/XSL/Transform"

is declared to get access to the XSLT elements, features and attributes. In the case of this application we are using XSLT for generating XHTML, therefore, the root element <html> is used.

Tables are created in the stylesheet to display the results from the previous step using <table>, <tr> and <th> tags. Suitable background colors and borders are defined.

The <xsl:for-each> element is used to loop through and select all elements from the specified node. The value of select attribute is an XPath expression, which navigates to the target node. The <xsl:value-of> extracts the value of an XML element and adds to the output of the transformation. In the Figure 16, a snippet of the stylesheet is shown; each value inside the name element is styled to be placed inside a cell in the table. The complete XSLT document is attached in the Appendix.

```
<html xsl:version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >
<head>
    <title>Query Result</title>
</head>
<body>

    <h2>Staff</h2>

    <table border="1">
        <tr bgcolor="#a0a0ff">
            <th>name</th>
            <th>position</th>
            <th>address</th>
            <th>email</th>
        </tr>

        <xsl:for-each select="People/People/Staff">
        <tr bgcolor="#ffffe9">
            <td>
                <xsl:value-of select="name"/>
            </td>
            <td>
```

Figure 16. XSLT stylesheet

This XSLT file is then referenced to the XML file (file resulted from the query). This transformation file is then executed and the XHTML document is obtained. The XHTML document thus produced contains the list of the students and staff, properly styled and well presented. This XHTML web-page content is an XML view over the database.

An XSLT compliant browser can also be used to nicely translate the XML file to XHTML. The complete XHTML document and its view over the brower is in the Appendix.

The steps for developing this application are shown in Figure 17.

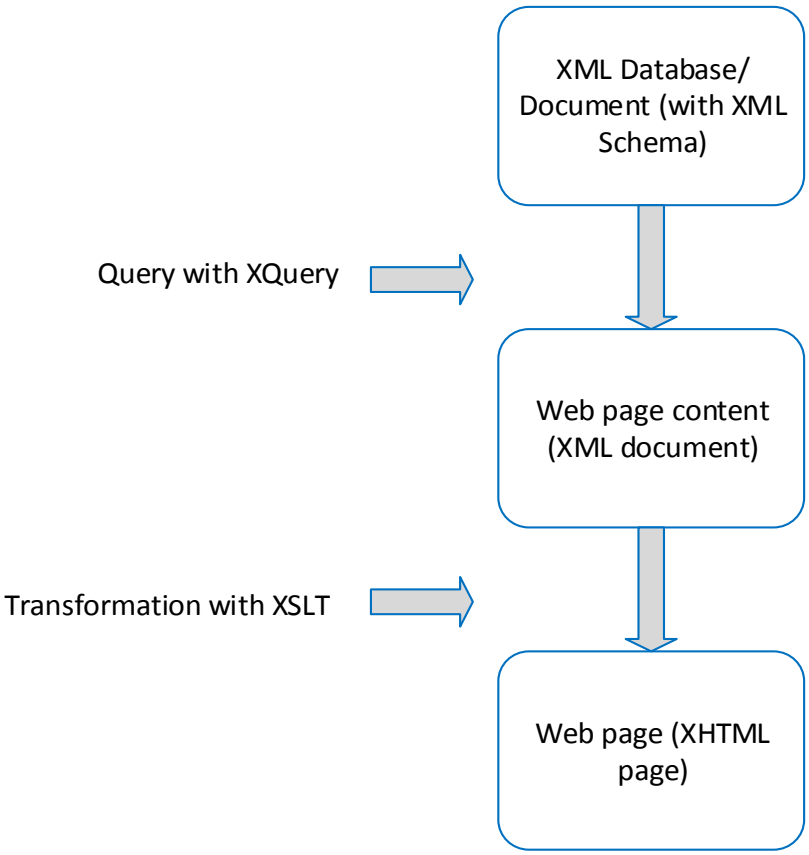


Figure 17. The complete step-by-step process

6 RESULTS

The main goal of this thesis was to develop an understanding of XML markup language and its possible application in database systems. Learning about different tools and technologies related to XML and developing a prototype application for Turku University of Applied Sciences was within the scope of this thesis. These goals were successfully achieved and a functional XML application was developed.

Altova's XML Spy was used to create this application. A sample database holding relevance to Turku University of Applied Sciences was created and constraints were defined with XML Schema Definition. XQuery was then used to create content of the web page as a view over the underlying XML database. Finally, XSLT was used to merge the content resulted from querying the XML database and presentation stylesheet to build web pages. These web pages were in the form of XHTML documents.

Few points that are needed to be considered when creating this application include:

- All the documents should be checked for well-formedness and validity errors.
- The documents must be referenced to one another accurately, for example the XML Schema file and the XML database file.

7 CONCLUSION

During the last half of the twentieth century there has been a tremendous increment in the amount of data. This has resulted in a greater demand for effectively storing, organizing, managing, efficiently accessing, extracting and sharing of data. With the advent of web during 1990s, there has been such a huge amount of data that managing those data using conventional tools has become impossible. It should not also be forgotten that most of the data present today are being viewed over various platforms, mobile handheld devices, etc. Therefore, as an effective solution to all these issues, Extensible Markup Language (XML) was developed.

This thesis provided an introduction to XML, its history and how different features of XML make it an alternative to conventional data models like entity-relationship database model. The application which was developed as a part of thesis made the readers familiar with the practical application of XML as a data model and how XML could be used to create meaningful data, where the tags could provide rich semantics to the data.

The thesis project has also demonstrated how the users can reinvent tags convenient for themselves, how the repurposing of the documents can be done to fit individual needs (HTML, XHTML, PDF, etc). Effectively querying the documents and extracting the information needed was also successfully demonstrated through this thesis project.

XML could be one of the alternatives to current database models. It could also serve as a one of the technologies upon which the future web could be built upon.

8 REFERENCES

[1] E. T. Ray, Learning XML:Guide to Creating Self-Describing Data, O'Reilly Media, 2001.

[2] E. R. Harold and W. S. Means, XML in a nutshell, Sebastopol: O'Reilly, 2001.

[3] A. Conrad and . D. Obasanjo, "XML & Relational Databases," 01 May 2003. [Online]. Available: http://www.drdobbs.com/database/xml-relational-databases/184405339. [Accessed 05 May 2014].

[4] B. Thuraisingham, XML Databases and the Semantic Web, Florida: CRC Press LLC, 2002.

[5] M. C. Daconta, L. J. Obrst and K. T. Smith, The Semantic Web, Indiana: Wiley, 2003.

[6] V. Geroimenko and C. Chen, Visualizing the Semantic Web:XML based internet and information visualization, London: Springer-Verlag London Limited, 2003.

[7] C. F. Goldfarb and P. Prescod, The XML handbook, Upper Saddle River, New Jersey: Prentice-Hall Inc., 2000.

[8] "XML Encoding," W3Schools, [Online]. Available: http://www.w3schools.com/xml/xml_encoding.asp. [Accessed 24 March 2014].

[9] "XML Tree," [Online]. Available: http://www.w3schools.com/xml/xml_tree.asp. [Accessed 25 March 2014].

[10] "XML Elements, Attributes, and Types (XML Designer)," Microsoft Developer Network, [Online]. Available: http://msdn.microsoft.com/en-

us/library/7f0tkwcx(v=vs.80).aspx. [Accessed 26 March 2014].

[11] "XML Attributes," [Online]. Available: http://www.w3schools.com/xml/xml_attributes.asp. [Accessed 26 March 2014].

[12] R. H. Elliotte, XML Bible, Foster City, CA: IDG Books Worldwide, Inc, 1999.

[13] "Extensible Markup Language (XML) 1.0 (Second Edition)," [Online]. Available: http://www.w3.org/TR/2000/REC-xml-20001006#dt-doctype. [Accessed 4 April 2014].

[14] "Advantages of using XML schemas instead of DTDs," IBM, [Online]. Available: http://publib.boulder.ibm.com/infocenter/db2luw/v8/index.jsp?topic=/com.ibm.db2.udb.doc/ad/c0008576.htm. [Accessed 9 April 2014].

[15] M. Jervis, "xml dtds xml schema," 26 November 2002. [Online]. Available: http://www.sitepoint.com/xml-dtds-xml-schema/. [Accessed 9 April 2014].

[16] B. McLaughlin, "Tip:Referencing XML Schema," 01 August 2001. [Online]. Available: http://www.ibm.com/developerworks/library/x-tipsch/x-tipsch-pdf.pdf. [Accessed 11 April 2014].

[17] "XML Path Language (XPath) 2.0 (Second Edition)," 14 December 2010. [Online]. Available: http://www.w3.org/TR/xpath20/. [Accessed 13 April 2014].

[18] "XPath Nodes," W3Schools, [Online]. Available: http://www.w3schools.com/xpath/xpath_nodes.asp. [Accessed 2014 April 13].

[19] P. Walmsley, XQuery, Sebastopol: O'Reilly Media, Inc., 2007.

[20] D. Tigwell, XSLT Mastering XML Transformations, Sebastopol: O'Reilly & Associates, Inc., 2008.

[21] "How XSLT Works," Lenz Consulting Group, Inc, [Online]. Available: http://lenzconsulting.com/how-xslt-works/. [Accessed 14 April 2014].

[22] "XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition)," [Online]. Available: http://www.w3.org/TR/xhtml1/#h-4.1 . [Accessed 2014 April 19].

[23] C. Musciano and B. Kennedy, HTML & XHTML: The Definitive Guide, 4th Edition, Sebastopol: O'Reilly & Associates, Inc., 2000..

[24] L. Underwood, "Why Switch to XHTML?," [Online]. Available: http://www.htmlgoodies.com/primers/html/article.php/3616146. [Accessed 2014 April 19].

[25] "Free Trial of Altova Software," [Online]. Available: http://www.altova.com/download-trial.html#. [Accessed 10 April 2014].

9 APPENDIX

Complete TUAS XML-database

```
<?xml version="1.0" encoding="UTF-8"?>
<TUAS xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="TUAS Schema.xsd">
        <Branches>
                <Joukahaisenkatu>
                        <People>
                                <Staff>
                                        <name>
                                                <first_name>Linda</first_name>
                                                <last_name>Nurkkala</last_name>
                                        </name>
                                        <position>Lecturer</position>
                                        <address>Joukahaisenkatu</address>
                                        <email>linda.nurkkala@turkuamk.fi</email>
                                </Staff>
                                <Student>
                                        <name>
                                                <first_name>Janne</first_name>
                                                <last_name>Pajunen</last_name>
                                        </name>
                                        <group>240S08</group>
                                        <address>Halinen</address>

                                        <email>janne.pajunen@students.turkuamk.fi
                                        </email>
                                </Student>
                        </People>
                        <Studies>
                                <Courses>
                                        <name>Physics</name>
                                        <credits>5</credits>
                                        <date>28.08.2013-15.12.2013</date>
                                        <instructor>Petteri Ahonen </instructor>
                                        <time>14:15-16:00</time>
                                        <place>C2025</place>
                                </Courses>
                                <Research>
                                        <topic>Semantic Web</topic>
                                        <content>Why is web semantic important?
                                        </content>
                                </Research>
                        </Studies>
                        <Others>Work placement opportunities</Others>
                </Joukahaisenkatu>
                <Sepänkatu>
                        <People>
                                <Staff>
```

```
                    <name>
                        <first_name>Maria</first_name>
                        <last_name>Ahtinen</last_name>
                    </name>
                    <position>Office staff</position>
                    <address>Sepänkatu</address>
                    <email>maria.ahtinen@turkuamk.fi</email>
                </Staff>
                <Student>
                    <name>
                        <first_name>Suvi</first_name>
                        <last_name>Niemi</last_name>
                    </name>
                    <group>NKITIS11</group>
                    <address>Nummi</address>
                    <email>suvi.niemi@students.turkuamk.fi</email>
                </Student>
            </People>
            <Studies>
                <Courses>
                    <name>Project Management</name>
                    <credits>5</credits>
                    <date>09.12.2013-14.04.2014</date>
                    <instructor>Jenni Niemi</instructor>
                    <time>12:15-14:00</time>
                    <place>C2042</place>
                </Courses>
                <Research>
                    <topic>Marketing strategies</topic>
                    <content>Marketing strategies</content>
                </Research>
            </Studies>
            <Others>Activities</Others>
        </Sepänkatu>
        <Salo>
            <People>
                <Staff>
                    <name>
                        <first_name>Harri</first_name>
                        <last_name>Rahkonen</last_name>
                    </name>
                    <position>Study counselor</position>
                    <address>Rantamaki</address>
                    <email>harri.rahkonen@turkuamk.fi</email>
                </Staff>

                <Student>
                    <name>
                        <first_name>Paula</first_name>
                        <last_name>Suominen</last_name>
                    </name>
                    <group>NINFOS11</group>
                    <address>Raunistula</address>
```

```
                                        <email>paula.suominen@students.turkuamk.fi
                                        </email>
                                </Student>
                        </People>
                        <Studies>
                                <Courses>
                                        <name>Linux Driver Development</name>
                                        <credits>4</credits>
                                        <date>22.04.2014-06.06.2014</date>
                                        <instructor>Ossi Virtanen</instructor>
                                        <time>14:15-16:00</time>
                                        <place>C2027</place>
                                </Courses>
                                <Research>
                                        <topic>Cyber crime</topic>
                                        <content>cyber crime</content>
                                </Research>
                        </Studies>
                        <Others>Activities</Others>
                </Salo>
                <Lemminkaisenkatu>
                        <People>
                                <Staff>
                                        <name>
                                                <first_name>Ari</first_name>
                                                <last_name>Huhtinen</last_name>
                                        </name>
                                        <position>Office staff</position>
                                        <address>Oriketo</address>
                                        <email>ari.huhtinen¤@turkuamk.fi</email>
                                </Staff>
                                <Student>
                                        <name>
                                                <first_name>Joonnas</first_name>
                                                <last_name>Rantamäki</last_name>
                                        </name>
                                        <group>ALIIBK11</group>
                                        <address>Kohmo</address>

        <email>joonas.rantamaki@students.turkuamk.fi</email>
                                </Student>
                        </People>
                        <Studies>
                                <Courses>
                                        <name>Accounting and Finance</name>
                                        <credits>5</credits>
                                        <date>01.01.2014-31.07.2014</date>
                                        <instructor>Matti Kempainen</instructor>
                                        <time>15:15-17:00</time>
                                        <place>B147</place>
                                </Courses>
                                <Research>
```

```
                                    <topic>Business Development</topic>
                                    <content>Business Development</content>
                                </Research>
                            </Studies>
                            <Others>Activities</Others>
                        </Lemminkaisenkatu>
                </Branches>
</TUAS>
```

XML Schema of TUAS XML document

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:element name="Salo">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element ref="People"/>
                                <xs:element ref="Studies"/>
                                <xs:element ref="Others"/>
                        </xs:sequence>
                </xs:complexType>
        </xs:element>
        <xs:element name="TUAS">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element ref="Branches"/>
                        </xs:sequence>
                </xs:complexType>
        </xs:element>
        <xs:element name="date">
                <xs:simpleType>
                        <xs:restriction base="xs:string">
                                <xs:enumeration value="01.01.2014-31.07.2014"/>
                                <xs:enumeration value="09.12.2013-14.04.2014"/>
                                <xs:enumeration value="22.04.2014-06.06.2014"/>
                                <xs:enumeration value="28.08.2013-15.12.2013"/>
                        </xs:restriction>
                </xs:simpleType>
        </xs:element>
        <xs:element name="name">
                <xs:complexType mixed="true">
                        <xs:sequence minOccurs="0">
                                <xs:element ref="first_name"/>
                                <xs:element ref="last_name"/>
                        </xs:sequence>
                </xs:complexType>
        </xs:element>
        <xs:element name="time">
                <xs:simpleType>
                        <xs:restriction base="xs:string">
                                <xs:enumeration value="12:15-14:00"/>
```

```
                        <xs:enumeration value="14:15-16:00"/>
                        <xs:enumeration value="15:15-17:00"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="Staff">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element ref="name"/>
                        <xs:element ref="position"/>
                        <xs:element ref="address"/>
                        <xs:element ref="email"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="email">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:enumeration value="ari.huhtinen¤@turkuamk.fi"/>
                        <xs:enumeration value="harri.rahkonen@turkuamk.fi"/>
                        <xs:enumeration
value="janne.pajunen@students.turkuamk.fi"/>
                        <xs:enumeration
value="joonas.rantamaki@students.turkuamk.fi"/>
                        <xs:enumeration value="linda.nurkkala@turkuamk.fi"/>
                        <xs:enumeration value="maria.ahtinen@turkuamk.fi"/>
                        <xs:enumeration
value="paula.suominen@students.turkuamk.fi"/>
                        <xs:enumeration
value="suvi.niemi@students.turkuamk.fi"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="group">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:enumeration value="240S08"/>
                        <xs:enumeration value="ALIIBK11"/>
                        <xs:enumeration value="NINFOS11"/>
                        <xs:enumeration value="NKITIS11"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="place">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:enumeration value="B147"/>
                        <xs:enumeration value="C2025"/>
                        <xs:enumeration value="C2027"/>
                        <xs:enumeration value="C2042"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
```

```
<xs:element name="topic">
        <xs:simpleType>
                <xs:restriction base="xs:string">
                        <xs:enumeration value="Business Development"/>
                        <xs:enumeration value="Cyber crime"/>
                        <xs:enumeration value="Marketing strategies"/>
                        <xs:enumeration value="Semantic Web"/>
                </xs:restriction>
        </xs:simpleType>
</xs:element>
<xs:element name="Others">
        <xs:simpleType>
                <xs:restriction base="xs:string">
                        <xs:enumeration value="Activities"/>
                        <xs:enumeration value="Work placement opportunities"/>
                </xs:restriction>
        </xs:simpleType>
</xs:element>
<xs:element name="People">
        <xs:complexType>
                <xs:sequence>
                        <xs:element ref="Staff"/>
                        <xs:element ref="Student"/>
                </xs:sequence>
        </xs:complexType>
</xs:element>
<xs:element name="Courses">
        <xs:complexType>
                <xs:sequence>
                        <xs:element ref="name"/>
                        <xs:element ref="credits"/>
                        <xs:element ref="date"/>
                        <xs:element ref="instructor"/>
                        <xs:element ref="time"/>
                        <xs:element ref="place"/>
                </xs:sequence>
        </xs:complexType>
</xs:element>
<xs:element name="Student">
        <xs:complexType>
                <xs:sequence>
                        <xs:element ref="name"/>
                        <xs:element ref="group"/>
                        <xs:element ref="address"/>
                        <xs:element ref="email"/>
                </xs:sequence>
        </xs:complexType>
</xs:element>
<xs:element name="Studies">
        <xs:complexType>
                <xs:sequence>
                        <xs:element ref="Courses"/>
                        <xs:element ref="Research"/>
```

```
                </xs:sequence>
        </xs:complexType>
</xs:element>
<xs:element name="address">
        <xs:simpleType>
                <xs:restriction base="xs:string">
                        <xs:enumeration value="Halinen"/>
                        <xs:enumeration value="Joukahaisenkatu"/>
                        <xs:enumeration value="Kohmo"/>
                        <xs:enumeration value="Nummi"/>
                        <xs:enumeration value="Oriketo"/>
                        <xs:enumeration value="Rantamaki"/>
                        <xs:enumeration value="Raunistula"/>
                        <xs:enumeration value="Sepänkatu"/>
                </xs:restriction>
        </xs:simpleType>
</xs:element>
<xs:element name="content">
        <xs:simpleType>
                <xs:restriction base="xs:string">
                        <xs:enumeration value="Business Development"/>
                        <xs:enumeration value="Marketing strategies"/>
                                <xs:enumeration value="Why is web semantics
important?"/>
                        <xs:enumeration value="cyber crime"/>
                </xs:restriction>
        </xs:simpleType>
</xs:element>
<xs:element name="credits">
        <xs:simpleType>
                <xs:restriction base="xs:byte">
                        <xs:enumeration value="4"/>
                        <xs:enumeration value="5"/>
                </xs:restriction>
        </xs:simpleType>
</xs:element>
<xs:element name="Branches">
        <xs:complexType>
                <xs:sequence>
                        <xs:element ref="Joukahaisenkatu"/>
                        <xs:element ref="Sepänkatu"/>
                        <xs:element ref="Salo"/>
                        <xs:element ref="Lemminkaisenkatu"/>
                </xs:sequence>
        </xs:complexType>
</xs:element>
<xs:element name="Research">
        <xs:complexType>
                <xs:sequence>
                        <xs:element ref="topic"/>
                        <xs:element ref="content"/>
                </xs:sequence>
        </xs:complexType>
```

```
</xs:element>
<xs:element name="position">
	<xs:simpleType>
		<xs:restriction base="xs:string">
			<xs:enumeration value="Lecturer"/>
			<xs:enumeration value="Office staff"/>
			<xs:enumeration value="Study counselor"/>
		</xs:restriction>
	</xs:simpleType>
</xs:element>
<xs:element name="last_name">
	<xs:simpleType>
		<xs:restriction base="xs:string">
			<xs:enumeration value="Ahtinen"/>
			<xs:enumeration value="Huhtinen"/>
			<xs:enumeration value="Niemi"/>
			<xs:enumeration value="Nurkkala"/>
			<xs:enumeration value="Pajunen"/>
			<xs:enumeration value="Rahkonen"/>
			<xs:enumeration value="Rantamäki"/>
			<xs:enumeration value="Suominen"/>
		</xs:restriction>
	</xs:simpleType>
</xs:element>
<xs:element name="Sepänkatu">
	<xs:complexType>
		<xs:sequence>
			<xs:element ref="People"/>
			<xs:element ref="Studies"/>
			<xs:element ref="Others"/>
		</xs:sequence>
	</xs:complexType>
</xs:element>
<xs:element name="first_name">
	<xs:simpleType>
		<xs:restriction base="xs:string">
			<xs:enumeration value="Ari"/>
			<xs:enumeration value="Harri"/>
			<xs:enumeration value="Janne"/>
			<xs:enumeration value="Joonnas"/>
			<xs:enumeration value="Linda"/>
			<xs:enumeration value="Maria"/>
			<xs:enumeration value="Paula"/>
			<xs:enumeration value="Suvi"/>
		</xs:restriction>
	</xs:simpleType>
</xs:element>
<xs:element name="instructor">
	<xs:simpleType>
		<xs:restriction base="xs:string">
			<xs:enumeration value="Jenni Niemi"/>
			<xs:enumeration value="Matti Kempainen"/>
			<xs:enumeration value="Ossi Virtanen"/>
```

```
                        <xs:enumeration value="Petteri Ahonen "/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="Joukahaisenkatu">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element ref="People"/>
                        <xs:element ref="Studies"/>
                        <xs:element ref="Others"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="Lemminkaisenkatu">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element ref="People"/>
                        <xs:element ref="Studies"/>
                        <xs:element ref="Others"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
</xs:schema>
```

XQuery Document

```
xquery version "1.0";

<People>
{
doc("TUAS xml.xml")/TUAS/Branches/*/People
}
</People>
```

XSLT Stylesheet

```
 <html xsl:version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >
<head>
            <title>Query Result</title>
</head>
<body>

        <h2>Staff</h2>

        <table border="1">
            <tr bgcolor="#a0a0ff">
                <th>name</th>
                <th>position</th>
```

```
                <th>address</th>
                <th>email</th>
            </tr>

            <xsl:for-each select="People/People/Staff">
            <tr bgcolor="#ffffe9">
                <td>
                    <xsl:value-of select="name"/>
                </td>
                <td>
                        <xsl:value-of select="position"/>
                </td>
                <td>
                        <xsl:value-of select="address"/>
                </td>
                <td>
                        <xsl:value-of select="email"/>
                </td>
            </tr>
            </xsl:for-each>
        </table>
            <h2>Student</h2>
            <table border="1">
            <tr bgcolor="#a0a0ff">
                <th>name</th>
                <th>group</th>
                <th>address</th>
                <th>email</th>
            </tr>

            <xsl:for-each select="People/People/Student">
            <tr bgcolor="#ffffe9">
                <td>
                    <xsl:value-of select="name"/>
                </td>
                <td>
                        <xsl:value-of select="group"/>
                </td>
                <td>
                        <xsl:value-of select="address"/>
                </td>
                <td>
                        <xsl:value-of select="email"/>
                </td>
            </tr>
            </xsl:for-each>
        </table>
</body>
</html>
```

XHTML view over browser

Staff

name	position	address	email
Linda Nurkkala	Lecturer	Joukahaisenkatu	linda.nurkkala@turkuamk.fi
Maria Ahtinen	Office staff	Sepänkatu	maria.ahtinen@turkuamk.fi
Harri Rahkonen	Study counselor	Rantamaki	harri.rahkonen@turkuamk.fi
Ari Huhtinen	Office staff	Oriketo	ari.huhtinen□@turkuamk.fi

Student

name	group	address	email
Janne Pajunen	240S08	Halinen	janne.pajunen@students.turkuamk.fi
Suvi Niemi	NKITIS11	Nummi	suvi.niemi@students.turkuamk.fi
Paula Suominen	NINFOS11	Raunistula	paula.suominen@students.turkuamk.fi
Joonnas Rantamäki	ALIIBK11	Kohmo	joonas.rantamaki@students.turkuamk.fi